12-2017

Datasets Used in Fifteen Years of Automated Requirements Traceability Research

Palak Sharma
ps2671@rit.edu

Datasets Used in Fifteen Years of Automated Requirements Traceability Research

by

Palak Sharma

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Software Engineering

Supervised by

Dr. Mehdi Mirakhorli

Department of Software Engineering

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

December 2017

The thesis "Datasets Used in Fifteen Years of Automated Requirements Traceability Research" by Palak Sharma has been examined and approved by the following Examination Committee:

Dr. Mehdi Mirakhorli
Assistant Professor
Thesis Committee Chair

Dr. J. Scott Hawker
Associate Professor
Graduate Program Director
Thesis Committee Member

Dedication

To my family for their love, sacrifices, prayers and assistance.

Special mention to my mother 'Anu', father 'Vijay' and husband 'Akshay' for supporting and encouraging me.

To professor Srikant Sangappa, whose lessons and advice positively contributed in my academic life and encouraged me to pursue an academic career.

Acknowledgments

I am extremely thankful to Dr. Mehdi Mirakhorli for the opportunity, guidance, patience and support while pursuing my Master's degree. I am also thankful to Dr. Hawker for his valuable comments on my thesis work. I also would like to thank Waleed Zogaan for his active collaboration in this project.Lastly, I would like to thank my research lab colleagues, Raghuram Gopalakrishnan, Joanna Cecilia Santos, Danielle Gonzalez and Adriana Sejfia for their valuable feedback and all the stimulating discussions.

Abstract

Datasets Used in Fifteen Years of Automated Requirements Traceability Research

Palak Sharma

Supervising Professor: Dr. Mehdi Mirakhorli

Datasets are crucial to advance automated software traceability research. Acquiring such datasets come in a high cost and require expert knowledge to manually collect and validate them. Obtaining such software development datasets has been one of the most frequently reported barrier for researchers in the software engineering domain in general. This problem is even more acute in field of requirement traceability, which plays crucial role in safety critical and highly regulated systems. Therefore, the main motivation behind this work is to analyze the current state of art of datasets used in the field of software traceability.

This work presents a first-of-its-kind literature study to review and assess the datasets that have been used in software traceability research over the last fifteen years. It articulates several attributes related to these datasets such as their characteristics, threats and diversity.

Firstly, 202 primary studies (refer Appendix A) were identified for purpose of this study, which were used to derive 73 unique datasets. These 73 datasets were studied in-depth and several attributes (size, type, domain, availability, artifacts) were extracted (refer Appendix B). Based on analysis of the primary studies, a threat to validity reference model,

tailored to Software traceability datasets was derived (refer to figure 4.4). Furthermore, to put some light upon the dataset diversity trend in the Software traceability community, a metric called Dataset Diversity Ratio was derived for 38 authors (refer to figure 4.5) who have published more than one publication in field of software traceability.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advances in the area of automated requirements traceability research rely on the availability of different types of *datasets*. *Training sets* are needed to train trace-algorithms based on Machine Learning (ML) techniques. For instance, researchers have used a labeled dataset of functional requirements and non-functional requirements to train classification techniques to create traceability links between quality attributes and requirements document, design models and source code [20, 62, 70, 75, 79]. *Validation sets* are needed to tune parameters of such trace-algorithms [18, 53, 62]. *Testing sets* are used to test the performance of trace-algorithms on unseen data. For instance, researchers have used datasets to evaluate the accuracy of trace-algorithms based on Information Retrieval (IR) that establish links between requirements and source code [27, 33, 40, 79, 82].

Obtaining such software development datasets has been one of the most frequently reported barriers for researchers in the software engineering domain in general [52, 77]. This problem is even more acute in the area of requirements traceability which is crucial in safety critical and highly regulated application domains [19]. Many of the publicly available open source systems are not representative of those domains and consequently may not be suitable to use for training, validation, or testing sets. Thus, it is important to explicitly state the threats to validity associated with the datasets so that research results are articulated with perspective to the underlying datasets.

Despite the crucial role of trace datasets, few efforts have been taken to understand the characteristics and limitations of the datasets in the area of requirements traceability [41] as well as the threats to validity associated with the results obtained with these datasets.

Similarly, few efforts have been taken to standardize how the datasets quality tracking and assurance should be implemented [78].

This study presents a systematic literature review (SLR) to assess the current state of software traceability datasets that have been used by researchers in the community over the past fifteen years. Specifically, the work investigates 1) the *characteristics* of those datasets, 2) the *threats to validity* associated with those datasets, and 3) the *diversity of datasets* used in the community. Specifically, this work provides new insights on the characteristics of datasets used in the Software traceability community, and reveals tacit information about a large number of datasets used in the community which can highlight the path for addressing the threats to validity of the research conducted in this area. The details regarding the studied papers and datasets are reported in Appendix A and Appendix B. For detailed information refer to the online repository[1].

Through a set of research questions, this study aims to explore the diversity, characteristics, and quality of the used datasets. The main objective of this study is to gain knowledge and have some insight about the used datasets domains, characteristics, and threats to validity in the studies related to software traceability; whether it is been used in training automated traceability approaches or as a case studies in the evaluation process for a proposed approaches.

The reminder of this thesis is organized as follows: Chapter 2 details the research questions answered in this work and the methodology followed to perform the SLR. Chapter 3 discusses related work. Chapter 4 describes the main findings. Chapter 5 lists possible threats to validity to this work. Chapter 6 concludes this work and outlines future research directions.

[1]Online appendix: `https://goo.gl/aUUFpb`

Chapter 2

Research method

To identify and collect the datasets used in the software traceability community, I conducted a systematic literature review (SLR) of all published full papers with *empirical* and *automated* software traceability *theme*. I followed the guidelines that were established by kitchenham et al. [46] for SLR in Software Engineering. In this work, traceability datasets refers to any form of data used by traceability researchers such as training set, testing sets, validation set, answer set, and case studies.

To ensure correctness of the data collected, an exhaustive peer review process was established at the start of the study and was carried out throughout the course of the study. Each of the phases are discussed in the following sections, section 2.1 : identifying *Research questions*, section 2.2 : deriving *Search strategy*, section 2.3 : creating *Inclusion and exclusion criteria*, section 2.4 : carrying out the *Study selection process* and section 2.5 : performing *Data extraction*. For the purpose of this study, each of these phases were divided into multiple tasks.

Following the defined peer review process, each of these tasks underwent several status stages, which were **To-Do** (task is created), **In-Review** (discussed in group meeting), **Completed** (task is approved in the group meeting). The group meetings comprised of two reviewers along with the author. This group of three reviewers (2 students and 1 professor) is referred to as **Review Team** for the purpose of this thesis.

2.1 Research questions

- *RQ1: What are the characteristics of traceability datasets?*

  This research question is further divided into multiple sub-research questions:

  - *RQ1.1: What are the source and target artifacts in traceability datasets?* I will collect the source and target types of the datasets and I will summarize the results by considering all traceability links types as bi-directional.
  - *RQ1.2: Which application domains are represented by traceability datasets?* I will identify the domain of each dataset and I will group the datasets based on their domains.
  - *RQ1.3: What is the size of traceability datasets?* To provide a standardized way of reporting size, I derived a metric called *trace space* that provides a proxy for the complexity of a dataset. Trace space, D, is defined as the product of the size of the source and the size of the target artifacts:

    $$D_{TraceSpace} = |D_{Source}| \times |D_{Target}| \tag{2.1}$$

    Note that trace space defines the maximum number of trace links between two artifacts.
  - *RQ1.4: What proportion of the traceability datasets is from industry, open-source projects, and student generated data?* I will use frequencies to answer this research question.
  - *RQ1.5: Are traceability datasets available for reuse?* I will investigate whether the datasets are available online.

- *RQ2: What are the threats to validity associated with traceability datasets?* I will categorize and summarize the threats to validity related to the usage of datasets, acknowledged or mitigated by the studied papers.

- *RQ3: Do we, as a community, strive for a diversity of traceability datasets?* This would provide an insight on whether the same datasets are used for all the research problems in hand or whether software traceability community seek to adopt new datasets for different research problems. For authors that have published more than one papers, I calculate a ratio that represents the diversity of the datasets that they use. The diversity ratio for author i, $DiversityRatio_i$, is defined as the number of unique datasets, $UniqueDataset_i$, divided by the total number of datasets, $TotalDatasets_i$, used across the publications of author i.

$$DiversityRatio_i = \frac{UniqueDataset_i}{TotalDatasets_i} \tag{2.2}$$

2.2 Search strategy

A search strategy is fundamental for any SLR to ensure that all relevant studies are considered for accurate conclusions [46,85]. The adopted search strategy consists of the following main elements: *search methods*, *search terms*, and *data sources*. I performed a preliminary search to retrieve existing literature reviews in the domain of software traceability. I found a few SLRs that are discussed in Section 3.1 but none of them address the research questions that are defined in this work.

The defined search strategy used both *manual* and *automatic* methods to ensure that I cover as many relevant venues and electronic data sources as possible [85]. In the *manual search*, I went through all papers published in the venues listed in Table 2.1. I built an initial list of relevant venues that was augmented by contacting traceability experts for suggestions on other related sources (conferences, journals, research groups active in this domain or individual papers). Table 2.1 contains venues that are considered high quality venues for software requirements (e.g., RE, TEFSE, and REJ), while other venues have a broader and generic theme (e.g., ICSE, TSE, and TOSEM). In the *automatic search*, a set of search terms was defined. I started with key terms used in traceability papers such as *requirements* and *traceability*. To be as generic as possible I expanded the search terms

to include *software traceability* which resulted in the final query as follows: *(software OR requirement) AND traceability*. I used the search terms during the automatic search to search in the electronic data sources listed in Table 2.2 by matching the terms with the title, keywords, and abstract of each paper.

Table 2.1: Venues used in the manual search phase.

Conferences	
ICSE	International Conference on Software Engineering
TEFSE	International Workshop on Traceability in Emerging Forms of Software Engineering
ASE	International Conference on Automated Software Engineering
ESEC	European Software Engineering Conference
FSE	International Symposium on Foundations of Software Engineering
SST	International Symposium on Software and Systems Traceability
RE	International Requirements Engineering Conference
REFSQ	International Working Conference on Requirements Engineering: Foundation for Software Quality
COMP-SAC	IEEE Computer Society International Conference on Computers, Software and Applications
ICSM	International Conference on Software Maintenance
MSR	International Conference on Mining Software Repositories
WICSA	Working IEEE/IFIP Conference on Software Architecture
ICPC	International Conference on Program Comprehension
ECSA	European Conference on Software Architectures
Journals	
EMSE	Empirical Software Engineering Journal
TSE	IEEE Trans. on Software Engineering Journal
ISSE	Innovations in Systems and Software Engineering Journal
SEP	Journal of Software: Evolution and Process
TOSEM	ACM Trans. on Software Engineering and Methodologies
REJ	Requirements Engineering Journal

Table 2.2: Databases used in the automatic search phase.

Database	**Web address**
ACM DL	http://portal.acm.org
IEEE Explorer	http://www.ieee.org/web/publications/ xplore
SpringerLink	http://www.springerlink.com
ScienceDirect	http://www.elsevier.com

2.3 Inclusion and exclusion criteria

The inclusion and exclusion criteria are specified in Table 2.3 and were applied at different stages to all of the retrieved studies (See Figure 2.1). To limit the scope of this SLR, I included all studies that were published between 2000 and 2016. All studies that have used datasets, case studies or empirical data to develop, validate, train, or test traceability techniques are selected as primary studies. Papers that only presented approaches or an

idea without empirical data-based validation were excluded. Main emphasis was on publications that focused on automated software traceability, therefore, publications related to models and processes of software traceability were excluded. In addition, I excluded short papers, workshops, and tool demonstration papers. Lastly, all duplicated studies found from different sources were identified and removed.

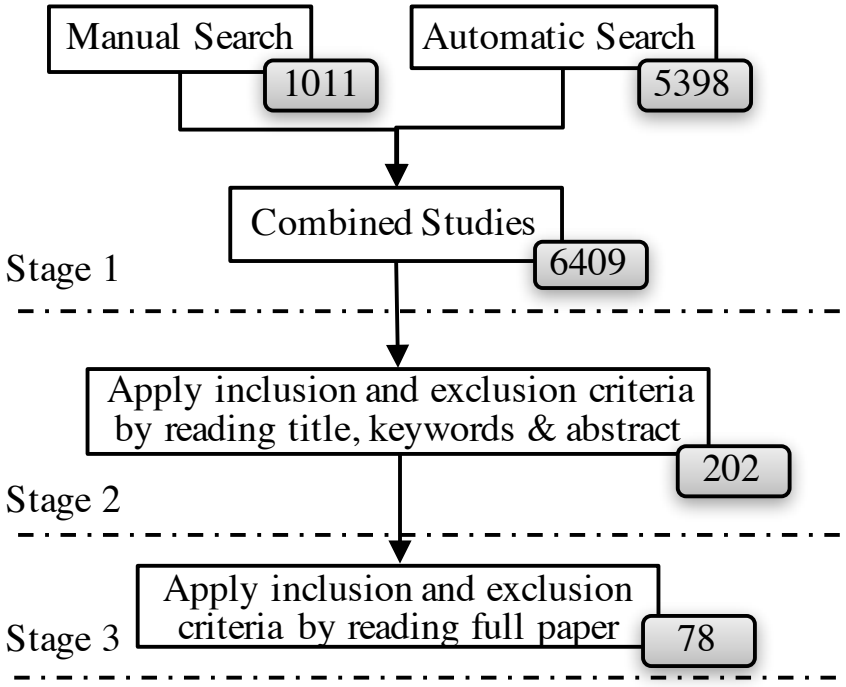

Figure 2.1: Search process stages.

Table 2.3: Inclusion and exclusion criteria.

Inclusion criteria	
I1	A Full paper.
I2	Focus on software (requirements) traceability.
I3	Proposed/used/evaluated an automated traceability technique.
I4	Used data-sets in their study.
Exclusion criteria	
E1	Position papers, short papers, tool demo papers, keynotes, reviews, tutorial summaries, and panel discussions.
E2	A study that is not written in English.
E3	Duplicated studies.
E4	No datasets or case studies.

2.4 Study selection process

Figure 2.1 shows the number of studies selected at each stage of the SLR. The initial search process resulted in 1011 and 5398 papers that were collected during the manual and automatic search, respectively. Because of a large number of retrieved papers (6409), I selected the first set of papers that could be relevant to this study by reading their title, keywords, and abstract [11]. From 6409 papers, 202 papers were selected as the primary studies. All these 202 papers were scrutinized by review team.For selecting a particular publication, the inclusion/exclusion criteria were applied and the rationale behind these decisions was documented. The rationale for including/excluding the studies were reassessed and discussed in separate review team discussions. If a publication satisfied all inclusion criteria, it was considered as a *primary study* and was included in SLR. The final list included 78 papers.

Table 2.4: Extracted dataset items.

Data item	**Related RQs**
The **Name** & **Traceability Artifacts**	RQ1, RQ1.1
The **Application Domain**	RQ1, RQ1.2
The **Size (trace space)**	RQ1.3, RQ2
The **Type** (private, student, or open source)	RQ1.4, RQ2
The **Availability** and link to the source	RQ1.5
The **Threats to validity** related to the datasets that are acknowledged or mitigated in the paper.	RQ2

2.5 Data extraction

In the data extraction phase, all information from the selected studies that was necessary to answer the identified research questions was extracted and analyzed. First, I extracted the basic information about the papers such as the list of the author(s), year of publication, title, venue, and publication type (refer to Appendix A). Then, through Review team meetings, dataset information was identified, extracted, and organized as used in each research paper. Table 2.4 provides an overview of the dataset attributes that were extracted from each paper and the research questions that require those attributes.

Chapter 3

Related Work

This section discusses the existing systematic literature review studies in domain of software traceability (Section 3.1).

3.1 SLRs in traceability

Several SLRs exist in field of software traceability [9], [64], [76], [19].

Borg et al. [9] conducted an SLR on Information Retrieval-based trace recovery that included 79 publications. However, this study mainly focused on the classification of publications based on the IR techniques used by the authors. In contrast, this study focuses on characterizing the datasets used in the domain of automated software traceability research. The authors discussed that most requirements document used by researchers had less than 500 requirements, and results were reported only using precision and recall. However, the SLR did not focus on studying the datasets used within the community, therefore, lacks insights in this regard.

Nair et al. [64] looked at 70 papers related to software traceability from the International Conference on Requirements Engineering (RE) and inspected various aspects of traceability. The study covered all studies published between 1993 and 2013 but the scope of the study is very limited, covering only papers published at RE. Regarding the datasets usage, the authors mentioned that out of 70 papers, 27 (38.7%) do not specify any details about the datasets. They reported a rising trend in the traceability field with increasing emphasis on quality of experimentation and academic-industrial partnership.

Santiago et al. [76] conducted an SLR on the impact of Model-Driven Engineering in traceability. Based on 157 studies that were published before 2011, they reported that storage, data related operations, and visualization are more widely studied aspects of traceability compared to exchange and analysis.

Liebchen and Shepperd [52] conducted a systematic review of 23 papers to study accuracy-based data quality. They concluded by stating that data quality should be taken into account while selecting a dataset. Another area of focus should be in identifying and correcting noisy datasets along with taking into account the impact of these noisy datasets on the results of the studies that use them.

Cleland-Huang et al. [19] analyzed the earlier and current trends in the field of software traceability. They pointed out some intriguing future research questions concerning the cost-effectiveness, trust, scalability, portability, ubiquity, and visualization aspects of traceability techniques. They reported that there is a lack of datasets that contain multiple artifact types (e.g., requirements, design, code, test cases, etc.), which in turn leads to limited studies in the direction of automation of traceability link evolution.

In this work, a systematic literature review is performed that focuses on the characteristics and threats related to traceability datasets.

Chapter 4

Results

This section summarizes the results of this study and answers the defined research questions. From all papers studied in this SLR, I identified 73 unique datasets. For each of these datasets, different attributes are collected as listed in Table 2.4.

4.1 RQ1: *What are the characteristics of traceability datasets?*

This question is investigated through five sub-questions described below. Each sub-question examines different characteristics of traceability datasets.

RQ1.1: What are the source and target artifacts in traceability datasets?

Figure 4.1 shows the types of artifacts covered by the *73* datasets. The inner layer represents the *source* artifact type and the outside layer represents the *target* artifact type. Artifacts from the inner and outside layers are colored identically when there is an association between them in the datasets. More details about these data points and their frequency can be found in Appendix B.

All artifacts that specify textual requirement documents related to a dataset such as high level, low level, functional, and non-functional requirements are grouped under the "Requirements" category. In a similar fashion, the "Code" category consists of Java Classes, Code, Methods, and Classes. The "Test Cases" category groups all non-code test documents whereas the "Unit test" category is composed of the actual code implementations. The category "Document" is composed of artifacts such as manuals and other pages that datasets are being traced to.

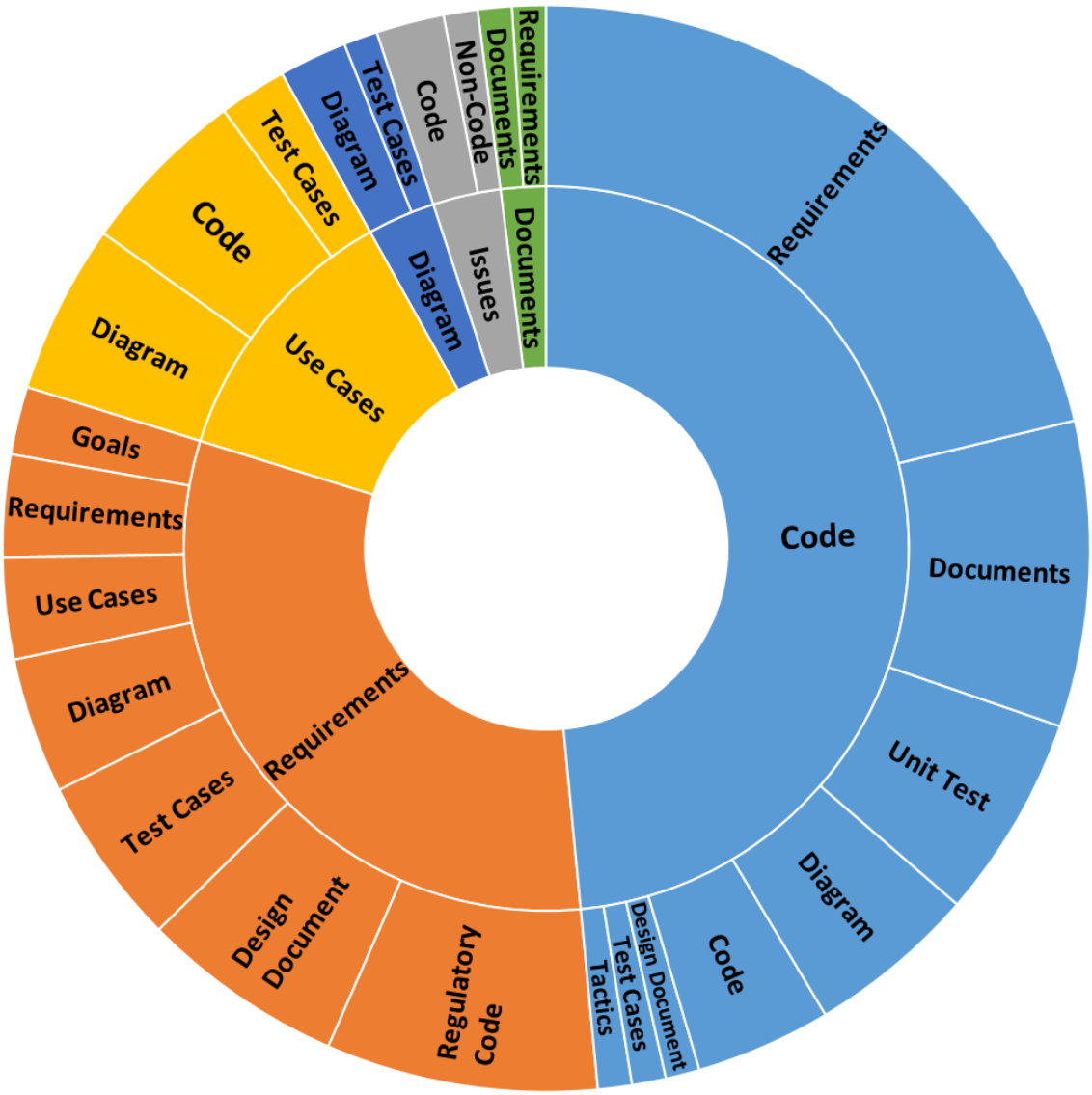

Figure 4.1: Common *Source* and *Target* Artifacts.

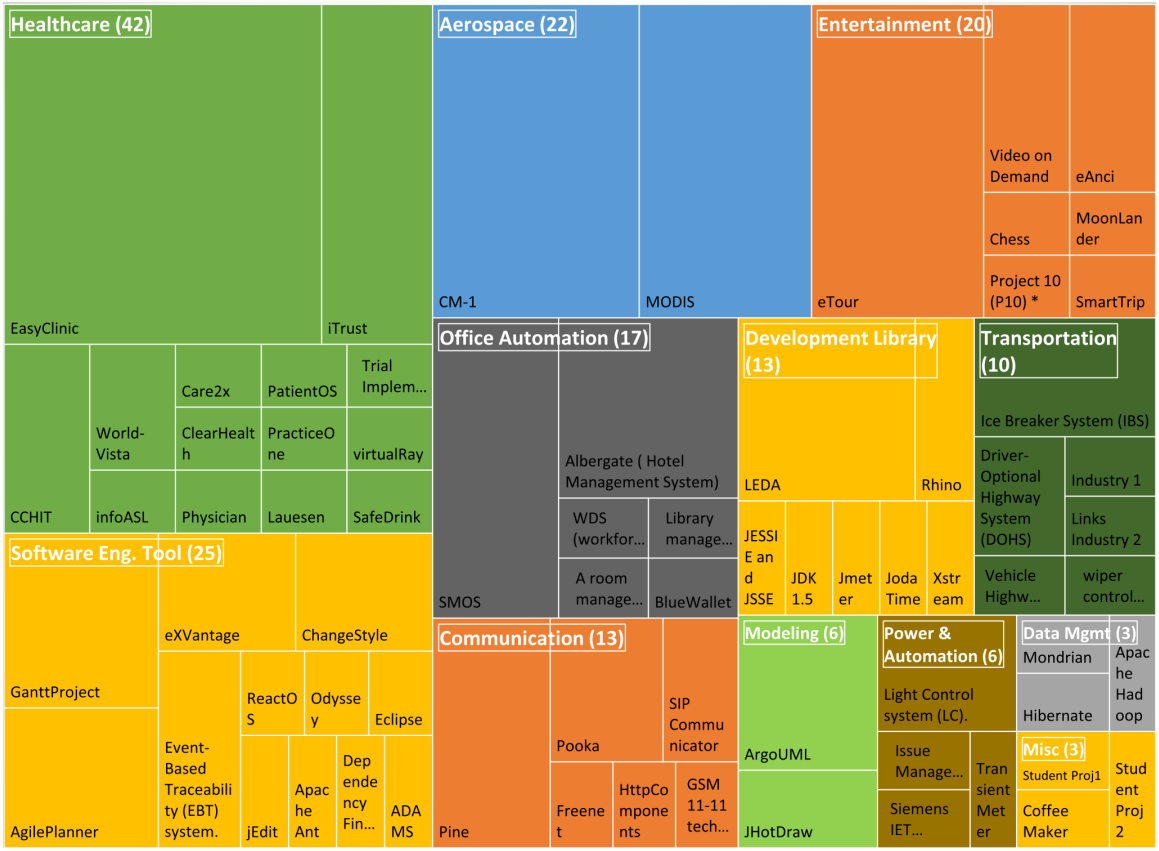

Figure 4.2: Dataset Domains and Frequency of Use.

As per the conducted analysis, the most frequently, datasets are used to study traces between Code to: Code (4), Unit Test (6) and other Non-Code artifacts such as Requirements (21), Documents (9), Diagram (5), Design Document (1), Test Cases (1) and Tactics (1). Another category of commonly considered artifacts was found to be between "Requirements" and other Non-Code artifacts such as Design Document (6), Goals (2), Regulatory Code (8), Test Cases (5), Diagram (4), Requirements (3), and Use Cases (3). Less studied artifacts were Use Cases, Issue Reports, Diagrams, and Documents.

RQ1.2: Which application domains are represented by traceability datasets?

The frequencies and domains of the datasets are shown as a heat-map in Figure 4.2. Each colored block refers to an application domain. The sub-areas within a block represent a particular dataset where the area represents the frequency of that dataset usage.

Healthcare is by far the most frequent domain for traceability datasets [3, 6, 7, 12–14, 18, 24, 25, 27–31, 37, 47, 51, 53–55, 57, 58, 65, 68, 69, 75, 78]. This is not surprising as traceability is crucial for safety critical and highly regulated domains [19]. Similarly, datasets from the *Aerospace* domain are frequently used by researchers [14, 28, 36, 41–44, 47, 50, 53, 54, 65, 67, 68, 78–80, 82, 86]. High proportion of datasets are also from the domains of *software engineering tools* [2, 15, 21, 26, 34, 63, 71–73, 81, 84, 86], *development libraries* [2, 4, 5, 15, 21, 22, 27, 41, 47, 48, 60, 83], and *entertainment* [6, 7, 14, 20, 23, 33, 35, 37, 38, 41, 45, 49, 50, 54, 56–58, 65, 69, 70, 78]. The majority of these systems are open source and available online which might explain their frequent usage by researchers. In addition, the researchers already serve as subject matter experts for some of these domains (e.g., software engineering tools or development libraries).

Industries such as *Power & Automation* have been used but less frequently [8, 16, 17, 32, 81, 86]. All except one of the datasets from this domain are closed source.

RQ1.3: What is the size of traceability datasets?

The conducted analysis shows that there is an enormous gap in size among datasets that have been used by researchers (Table 4.1). The minimum trace space size is from the industrial datasets and it is 42 while the maximum one is over 29 million and it is a software system in the *power and automation* domain, containing 4845 issue reports and 6104 non-code artifacts [8]. The median of the trace space size among the three different datasets sources is relatively small.

Table 4.1: Datasets' trace space statistics.

Statistics	OSS	Private/Industrial	University/Students
Minimum	264	42	50
First Quartile	870	1082	1515
Median	2028	2926	5135
Third Quartile	6956	131690	15472
Maximum	49810	29573880	390978

RQ1.4: What proportion of the traceability datasets is from industry, open-source projects, and student generated data?

As shown in Figure 4.3 there is a fair distribution among the different types of sources:

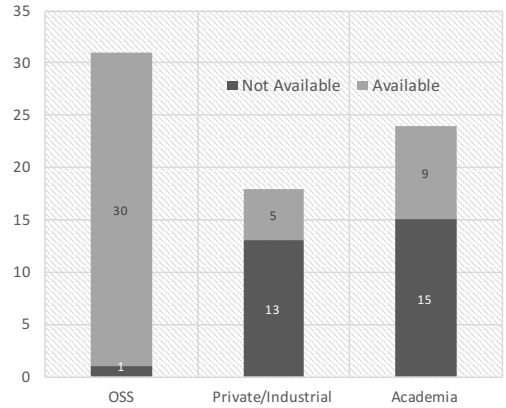

Figure 4.3: Source and availability of datasets.

31 datasets are open-source software (OSS), 24 datasets come from academia (e.g. student projects), and 18 datasets are industrial projects.

RQ1.5: Are traceability datasets available for reuse?

Figure 4.3 shows that 39.7% of the datasets (29 out of 73) are not available. Almost all of the OSS datasets are available. The majority of the industrial datasets (13 out of 18) are not available. The majority of the datasets coming from academia are not available (15 out of 24).

4.2 RQ2: *What are the threats to validity associated with traceability datasets?*

There are several threats associated with any form of empirical research involving data and reasoning on data. The results of such studies are typically limited to the dataset used, methods and context of the study. In this section a threat reference model is derived based on the threats identified and discussed by numerous researchers in the domain of software traceability. Among the 78 papers included in this SLR, 40% did not include a section related to the threats to validity nor discussed such concern while heavily relying on datasets to make research conclusion. 6% of the papers did have a threats to validity section, but did not identify any threats related to the usage of the data in their studies. Lastly, 54% of the papers discussed the threats to validity of their research related to the

usage of datasets.

All the threats to validity related to the datasets were extracted and manually grouped. Note that this includes all threats that are discussed by the authors of the respective publications, which means that they were not necessarily mitigated. Based on these threat categorization, a threat to validity reference model was derived as represented in figure 4.4. The threats to validitys associated with datasets are as follows:

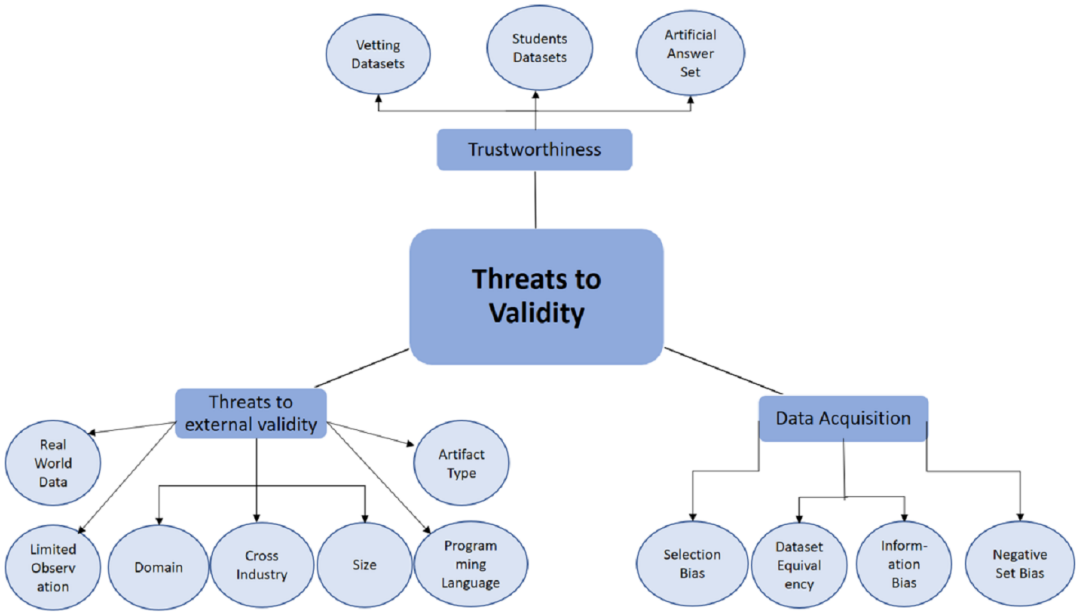

Figure 4.4: Threat Reference Model for Datasets

- *Trustworthiness*
  - *Artificial AnswerSet:* This threat is concerned with how answersets are created [15, 24, 40, 44, 53, 55, 62, 74]. Often the trustworthiness threat is not mitigated as the answersets are established by students rather than the original developers.
  - *Students Dataset*: This threat concerns dataset that are developed by students [37].
  - *Vetting Datasets*: This threat concerns datasets, particularly answersets, that are not vetted nor peer-reviewed [44, 62].

- *Threats to external validity*
  - *Real-World Data*: This threat is concerned with whether the datasets are representative of industrial projects [1, 3, 6, 7, 14, 15, 18, 24, 31, 33, 34, 37, 44, 45, 50, 54, 58, 59, 61, 62, 65, 67, 69].
  - *Limited Observations*: This threat is concerned with whether a limited number of case studies are used to validate the results [1, 3, 22, 29, 36, 39, 44, 45, 47, 53, 55, 58, 62, 67, 68, 86].
  - *Domain*: This threat is a concern when all datasets belong to the same application domain [15, 39, 40, 50, 61, 75] or when the number of datasets is insufficient to generalize the conclusions for a particular domain.
  - *Cross Industry*: When an industrial dataset is used, this threat is concerned with whether the results are applicable to other industrial systems [66].
  - *Size*: This threat is related to the small size of datasets, impacting the generalizability of the results [2, 3, 6, 14, 24, 34, 44, 45, 50, 54, 57, 61, 67, 72, 75].
  - *Programming Language*: This threat is a concern when datasets are in a specific programming language [12, 13, 37].
  - *Artifact Type*: This threat is concerned with the diversity of the type of artifacts available for the datasets (e.g., requirements, test cases, etc.) [47, 75].
- *Data Acquisition*
  - *Selection bias*: When datasets are not representative of the intended population (cherry picking) [18] or do not fit the problem [10, 18, 33, 36, 49]. For instance, this happens when a dataset from a non-safety critical project is used for a safety critical research study.
  - *Dataset-Equivalency*: This threat to validity concerns cases where researchers compare certain characteristics of their datasets with datasets used by previous researchers to justify the adequacy of the selected datasets [3, 33].

- *Information bias*: Accuracy of the automatically generated datasets; misclassification and labeling of the data to be used [12, 12, 13, 49].
- *Negative Set Bias*: Rich and unbiased selection of negative cases in training data, a common threat in classification problems [62].

The scope of this work was limited to gathering and categorizing all the possible threats related to datasets, as mentioned by researchers in field of Software Traceability. Though, the threat reference model (refer to figure 4.4) acknowledges the dataset threats but it lacks the mitigation strategies for these threats. Adding mitigation strategies to the threat reference model requires further analysis and validation, which is a significantly important future work. Furthermore, the threat reference model can be made more comprehensive by including these mitigation strategies along with the respective threats.

4.3 RQ3: *Do we, as a community, strive for a diversity of traceability datasets?*

To answer this research question, I studied the diversity of datasets used by authors across different research papers. First, all authors who have published more than one traceability paper were identified. This took the author count from 128 served on the 78 studied papers to 38 authors who have published more than one paper. For each of these authors, the diversity metric defined in Equation 2.2 was calculated.

Figure 4.5 shows the results. The X-axis represents the total number of datasets used by each author. The Y-axis represents the total number of papers from each author in this SLR. The Z-axis represents the diversity ratio for the datasets used by the authors. Each vertical drop-line corresponds to one of the 38 authors. 12 authors from these 38, have a diversity ratio of 70% and above, 27 authors have a diversity rate of 50% and above, and lastly, 11 authors have a diversity rate below 50%.

An interesting observation from the figure is that in general authors with low number of datasets and low number of papers have a higher diversity ratio. One of the authors with

high number of datasets and high number of papers has a high diversity rate. This example highlights individual effort in seeking diverse datasets for development and evaluation of various traceability solutions.

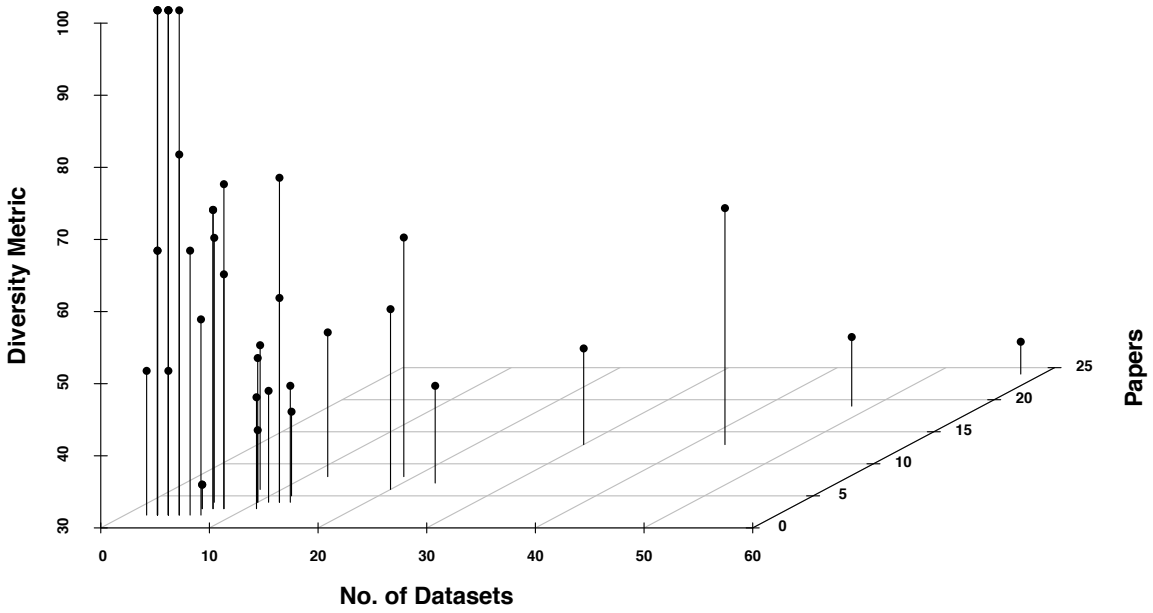

Figure 4.5: Authors and their Dataset Diversity.

Chapter 5

Threats to Validity

There are two main threats to validity impacting this SLR: bias in the study selection and bias in the extraction of data. Study selection can be dependent on the individuals reviewing the papers and therefore, the researchers knowledge in the domain could affect the results of this SLR. To minimize such bias, a restrict review protocol was established in which two students (including the author) selected the papers and documented their rationale for including or excluding the paper from the study. Then, the decisions were reviewed in follow-up Review team discussions. The search strategy includes both automatic and manual approaches. The automatic search relies on the title, abstract, and keywords of papers. The manual search was conducted to complement the automated search and reduce the chances of missing relevant papers from known traceability venues.

To minimize the threat related to the data extraction process, Google Doc spreadsheets were used to collect data from each publications (refer : Appendix A) and to document agreements and disagreements within the review team. To reduce errors while collecting information for each dataset, several publications were used that utilize the same datasets as a sanity check. In case a dataset was available online, I reviewed the dataset itself to get the desired attribute information.

Chapter 6

Conclusion

What can be learned from this study? First, this study highlights the detailed characteristics of the datasets used in the domain of software traceability. This provides an in-depth understanding of the current state of the datasets used in the community, their strength, and shortcomings. Such novel knowledge draws the attention of Software Traceability community to the areas that can improve rigorousness of evaluation and practicality of research.

This study can be used as a guideline for the researchers to select datasets based on their research needs and the characteristics of the datasets. For instance, for each of these datasets (reported in Table 6.1), Appendix B provides a link to the dataset, along with other meta-data details associated with it, such as previous studies that used it, threats to validity associated with it, trace space, and other characteristics of the dataset.

Furthermore, this study makes the tacit community wide threats related to the datasets explicit. The threats reference model (reported in figure 4.4) can be used as a guideline to (1) understand threats related to dataset and to (2) select datasets in-order to mitigate these threats. This will help the researchers in traceability community to not only better understand the strengths and weaknesses of the empirical foundations, but also, it will assist in making more informed decisions regarding dataset selection.

Table 6.1: Datasets' information and the studied papers which used the datasets.

Paper	DataSet Name	General Domain	Avail	URL	OSS	Ind.	Aca.
[63]	Odyssey	SE Tool	N	NA	N	N	Y
[74]	Mondrian	Data Management	Y	https://goo.gl/uQRIRC	Y	N	N
[71–74]	AgilePlanner	SE Tool	Y	https://goo.gl/nROJEo	Y	N	N
[84]	JESSIE and JSSE	Development Library	Y	https://goo.gl/gqk98Q	Y	N	N
[21]	Eclipse	SE Tool	Y	http://www.eclipse.org	Y	N	N
[2, 21]	Rhino	Development Library	Y	https://goo.gl/YemfFG	Y	N	N
[15]	JDK 1.5	Development Library	Y	http://goo.gl/mznQqF	Y	N	N
[15, 71–73]	ArgoUML	Modeling	Y	http://goo.gl/G9PzHC	Y	N	N
[15]	Freenet	Communication	Y	https://goo.gl/0I8XYi	Y	N	N
[15]	Jmeter	Development Library	Y	http://jmeter.apache.org/	Y	N	N
[14, 24, 47, 54, 79]	Pine	Communication	N	NA	N	N	Y
[1–3]	Pooka	Communication	Y	https://goo.gl/nfXrrg	Y	N	N
[2]	jEdit	SE Tool	Y	http://www.jedit.org/	Y	N	N
[22]	Joda Time	Development Library	Y	https://goo.gl/sNxXoN	Y	N	N
[22]	HttpComponents	Communication	Y	https://goo.gl/cIqNuX	Y	N	N
[22]	Hibernate	Data Management	Y	https://goo.gl/f3n52Z	Y	N	N
[22]	Xstream	Development Library	Y	https://goo.gl/e447m6	Y	N	N
[34, 38, 44, 49, 78]	GanttProject	SE Tool	Y	https://goo.gl/MlJx4D	Y	N	N
[38, 49]	JHotDraw	Modeling	Y	https://goo.gl/sNzUkb/	Y	N	N
[71–73]	eXVantage	SE Tool	N	NA	N	Y	N
[41, 47, 48]	ChangeStyle	SE Tool	N	NA	N	N	Y
[34]	ReactOS	SE Tool	Y	https://goo.gl/4xv4Qc	Y	N	N
[72, 81, 86]	EBT system.	SE Tool	N	NA	N	N	Y
[86]	SE450 Projects	Miscellenous	N	NA	N	N	Y
[73]	Apache Ant	SE Tool	Y	https://goo.gl/uJC8wx	Y	N	N
[73]	Dependency Finder	SE Tool	Y	https://goo.gl/fgElsu	Y	N	N
[27]	ADAMS	SE Tool	N	NA	N	N	Y
[62]	Apache Hadoop	Data Management	Y	https://goo.gl/zLI9ZW	Y	N	N
[4, 5, 5, 27, 60, 83]	LEDA	Development Library	Y	http://goo.gl/RbyyoM	N	N	Y
[18, 53, 75]	CCHIT	Healthcare	Y	https://goo.gl/3KPkmO	Y	N	N
[18, 75]	World- Vista	Healthcare	Y	https://goo.gl/Sv8DDK	Y	N	N
[6]	infoASL	Healthcare	N	NA	N	N	Y
[3, 18, 53, 54, 58, 65, 68]	iTrust	Healthcare	Y	https://goo.gl/Jz7HQn	N	N	Y
[18]	Care2x	Healthcare	Y	https://goo.gl/2VIdRL	Y	N	N
[18]	ClearHealth	Healthcare	Y	https://goo.gl/xLS2qM	Y	N	N
[18]	Physician	Healthcare	N	NA	N	Y	N
[18]	PatientOS	Healthcare	Y	https://goo.gl/u7lqPZ	Y	N	N
[18]	Trial Implementations	Healthcare	Y	https://goo.gl/vLHwgR	N	Y	N
[18]	PracticeOne	Healthcare	Y	https://goo.gl/W9sH3q	N	Y	N
[18]	Lauesen	Healthcare	Y	https://goo.gl/7Hblkc	N	N	Y
[6]	virtualRay	Healthcare	N	NA	N	N	Y
[78]	WV-CCHIT	Healthcare	Y	https://goo.gl/hbwgsz	Y	N	N
[57]	SafeDrink	Healthcare	N	NA	N	Y	N
[6, 7, 12–14, 24, 25, 27–31, 37, 47, 53–55, 68, 69, 78]	EasyClinic	Healthcare	Y	https://goo.gl/1RxkzC	N	N	Y
[14, 36, 41, 44, 47, 53, 65, 78–80, 82, 86]	CM-1	Aerospace	Y	https://goo.gl/KlGwSh	N	Y	N
[14, 28, 41–43, 50, 54, 67, 68, 80]	MODIS	Aerospace	Y	https://goo.gl/AzOnSm	N	Y	N
[38]	Chess	Entertainment	N	NA	Y	N	N
[4, 5, 5, 23, 60, 78]	Albergate	Office Automation	Y	https://goo.gl/Vc0NGT	N	N	Y
[61]	CoffeeMaker	Miscellenous	Y	https://goo.gl/mzoTtr	N	N	Y
[35, 38, 49]	Video on Demand	Entertainment	Y	https://goo.gl/oEZPsW/	Y	N	N
[70]	Project 10 (P10) *	Entertainment	Y	https://goo.gl/dRQwRt	Y	N	N
[50]	MoonLander	Entertainment	N	NA	N	N	Y
[16, 20, 81, 86]	Ice Breaker System	Transportation	N	NA	N	N	Y
[51]	Vehicle Highway System	Transportation	N	NA	N	Y	N
[39, 40]	Driver-Optional Highway System (DOHS)	Transportation	N	NA	N	Y	N
[53]	Industry 1	Transportation	N	NA	N	Y	N
[53]	Links Industry 2	Transportation	N	NA	N	Y	N
[56]	Wiper control system	Transportation	N	NA	N	Y	N
[1, 2]	SIP Communicator	Communication	Y	https://goo.gl/CulKDb	Y	N	N
[10]	GSM 11-11	Communication	Y	https://goo.gl/7BsrCW	N	Y	N
[8]	Issue Management System (IMS)	power and automation	N	NA	N	Y	N
[17]	Siemens IET document	power and automation	N	NA	N	Y	N
[32]	Transient Meter	power and automation	N	NA	N	N	Y
[66]	Workforce development	Office Automation	N	NA	N	Y	N
[57]	BlueWallet	Others	N	NA	N	Y	N
[16, 81, 86]	Light Control system	power and automation	N	NA	N	N	Y
[14, 20, 23, 33, 37, 41, 54, 58, 65, 69]	eTour	Entertainment	Y	https://goo.gl/bYLzQD	N	N	Y
[45]	Room management	Office Automation	N	NA	N	N	Y
[57]	SmartTrip	Entertainment	N	NA	N	Y	N
[6, 37, 78]	eAnci	Entertainment	Y	https://goo.gl/9sVdFu	N	N	Y
[56]	Library system	Office Automation	N	NA	N	N	Y
[6, 7, 23, 33, 37, 69, 78]	SMOS	Office Automation	Y	https://goo.gl/UALgaf	N	N	Y
[26]	7 Student Projects	Miscellenous	N	NA	N	N	Y

Avail: If dataset is available; OSS: Open Source Project; Ind.: Industrial Project; Aca.: Academic Project

Appendix A

Publication Characteristics

Table A.1: Publication Characteristics

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
3	Requirements traceability in automated test generation: application to smart card software validation	Communication	1	Req	Test Cases	42	Industrial	GSM 11-11 technical specification	1
4	Assessing IR-based traceability recovery tools through controlled experiments	Healthcare	1	SW Artifacts	code	2370	University	EasyClinic	1
6	IR-Based Traceability Recovery Processes: An Empirical Comparison of ”One-Shot” and Incremental Processes	Healthcare	1	SW Artifacts	code	2370	University	EasyClinic	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
8	Incremental Latent Semantic Indexing for Automatic Traceability Link Evolution Management	Development Library	1	SW Artifacts	code	1668244	University	1) LEDA 3.42) LEDA 3.4.1	2
12	ArchTrace: Policy-Based Support for Managing Evolving Architecture-to-Implementation Traceability Links	software Eng. Tool	1	Req	Code	N/A	OSS	Odyssey	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
15	Rule-Based Maintenance of Post-Requirements Traceability Relations	Office Automation, Transportation	2	Req (UML)	Code	N/A	University	1) library management system. 2) wiper control system for a car,	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
19	Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability	Transportation, Power & Automation, Software Eng. Tool	3	Req	UML	17100	OSS	1) Ice Breaker System (IBS). 2)Event-Based Traceability (EBT) system. 3)Light Control-system (LC).	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
20	Recovering traceability links in software artifact management systems using information retrieval methods	Miscellenous	1	SW Artifacts	Code	14419	University	17 Student SW	17

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
26	Recovering Traceability Links between Code and Documentation	Development Library, Office Automation	2	SW Artifacts	Code	19264	University	1) LEDA (Library of Efficient Data types and Algorithms), 2) Albergate (Hotel Management System).	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
32	Recovering traceability links between unit tests and classes under test: An improved method	Software Eng. Tool, Data Mgmt	2	Unit Tests	Classes	32304	Industrial, OSS	1) Agile-Planner. 2) Mondrian	2
34	Traceability for the maintenance of secure software	Development Library	1	Design	Code	133	OSS	JESSIE and JSSE	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
35	Information Retrieval Models for Recovering Traceability Links between Code and Documentation	Development Library, Office Automation	2	SW Artifacts	Code	19264	University	1) LEDA (Library of Efficient Data types and Algorithms). 2) Albergate (Hotel Management System).	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
36	Source Code Indexing for Automated Tracing	Healthcare, Entertainment	2	Req	Code	15316	University	1) eTour. 2)iTrust	2
39	Recovering Traceability Links between Source Code and Fixed Bugs via Patch Analysis	Software Eng. Tool, Development Library	2	Bug Reports	Code	467938	OSS	1) Eclipse. 2) Rhino	2
42	Combining textual and structural analysis of software artifacts for traceability link recovery	Miscellenous	1	Req	Code	272	University	1) CoffeeMaker.	1
45	Ontology-Based Trace Retrieval	Transportation	1	Req	Design	359566	Industrial	1) Vehicle Highway System	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
46	REquirements TRacing On target (RETRO) Enhanced with an Automated Thesaurus Builder: An Empirical Study	Aerospace	1	Req	Req	931	Industrial	1) MODIS	1
49	Adaptive User Feedback for IR-Based Traceability Recovery	Healthcare, Aerospace	2	SW Artifacts	SW Artifacts	7793	Industrial,University	1) EasyClinic. 2) i-Trust 3) Modis	3
54	Towards an intelligent domain-specific traceability solution	Transportation	1	Req	Req	794475	Industrial	Driver-Optional Highway System (DOHS)	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
56	Improving automated documentation to code traceability by combining retrieval techniques	Development Library, Modeling, Communication	3	Req	code	526992	OSS, Industrial	1) JDK 1.5 2) Argu-UML3) Freenet4) Jmeter	4
60	Automating Requirements Traceability: Beyond the Record & Replay Paradigm	Entertainment	1	Req	Code	210	OSS	Video on Demand	1
62	Traceability-enabled refactoring for managing just-in-time requirements	Office Automation	1	Req	Code	2500	Industrial	WDS	1
63	Supporting traceability through affinity mining	Aerospace	1	Req	Req	51700	Industrial	CM-1	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
65	Foundations for an expert system in domain-specific traceability	Transportation	1	Req	Req	794475	Industrial	Driver-Optional Highway System (DOHS)	1
66	Application of reinforcement learning to requirements engineering: requirements tracing	Communication, Aerospace	2	Req	Req	3665	University, Industrial	1. Pine2. CM-1	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
90	Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links	Software Eng. Tool, Development Library,Communication	3	Req	Code	225448	OSS	1) jEdit2) Pooka3) Rhino4) SIP Communicator	4

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
93	Recovering traceability links between an API and its learning resources	Development Library, Communication, Data Mgmt	3	Doc	Code	3581	OSS	1) Joda Time2) Http-Components3) Hibernate4) Xstream	4
100	Goal-centric traceability for managing non-functional requirements	Transportation	1	Req	Code	13500	University	1) Ice Breaker System (IBS).	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
101	Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing	Development Library, Office Automation	2	Document	Source Code	41488	University	1) LEDA library, 2) Albergate	2
104	Enhancing Software Traceability by Automatically Expanding Corpora with Relevant Documentation	Office Automation, Entertainment	2	Req	Code	14280	OSS, University	1) Albergate2) eTour3) SMOS	3
106	Do data dependencies in source code complement call dependencies for understanding requirements traceability?	Entertainment, Software Eng. Tool, Modeling	3	Req	Code - methods	238350	OSS	1) VideoOnDemand (VoD),2)GanttProject3) JHotDraw	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
107	An empirical study on requirements traceability using eye-tracking	Healthcare, Communication	2	Req	source code	45230	University, OSS	1) iTrust 2) Pooka	2
109	SCOTCH: Test-to-code traceability using slicing and conceptual coupling	Modeling, Software Eng. Tool	2	Unit tests	Code Classes	122734	OSS, Industrial	1) AgilePlanner, 2) eXVantage, 3) ArgoUML.	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
111	On integrating orthogonal information retrieval methods to improve traceability recovery	Healthcare, Office Automation, Entertainment	3	Use cases, UML Diagrams, Test Cases	Code Classes	28618	University	1) eAnci,2) EasyClinic (English and Italian versions), 3) eTour (English and Italian versions), 4) SMOS	4

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
114	Incremental Approach and User Feedbacks: a Silver Bullet for Traceability Recovery.	Aerospace, healthcare	2	artifacts	artifacts	5741	OSS, Industrial, University	1) MODIS2) Easy-Clinic	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
115	Enhancing an Artefact Management System with Traceability Recovery Features	Healthcare	1	artifacts :requirement, design,and testing documents, as well as code components.	artifacts :requirement, design,and testing documents, as well as code components.	3861	University	EasyClinic	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
120	Inherent characteristics of traceability artifacts less is more.	Aerospace, Entertainment, Software Eng. Tool	3	HL Req	LL Req	527	OSS, Industrial, University	1) CM-1,2) MODIS, 3) ChangeStyle, and 4) Etour.	4
121	Text Mining Support for Software Requirements: Traceability Assurance	Entertainment	1	Functional Requirements	Non Functional Requirements	570	OSS	Project 10 (P10)	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
122	Proximity-based traceability: An empirical validation using ranked retrieval and set-based measures	Aerospace, Communication, Software Eng. Tool, Healthcare	4	Text based artifacts	Text based artifacts	9637	Industrial,OSS, University	1) NASA CM 12) Pine3) ChangeStyle 4) EasyClinic	4
123	Toward Automating Requirements Satisfaction Assessment.	Aerospace, Software Eng. Tool	2	Req	Low level element (design)	53026	OSS, Industrial	1) NASA CM-12) GanttProject	2
124	Tracing requirements to defect reports: an application of information retrieval techniques	Aerospace	1	Req	Defect Reports	2540	OSS	CM-1	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
125	Baselines in requirements tracing	Aerospace	1	Req	artifacts	52631	OSS	1) NASA Moderate Resolution Imaging Spectroradiometer (MODIS) documents2) CM1	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
126	Helping Analysts Trace Requirements: An Objective Look	Aerospace	1	Req	artifacts	931	OSS	NASA Moderate Resolution Imaging Spectroradiometer (MODIS) documents	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
127	Improving Requirements Tracing via Information Retrieval	Aerospace	1	Req	Level 1A (L1A) and Geolocation Processing Software Requirements Specification	1050	OSS	NASA Moderate Resolution Imaging Spectroradiometer (MODIS) documents	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
131	Automated extraction and visualization of quality concerns from requirements specifications	Healthcare	1	quality concern goals	Requirements	19710	OSS	1) Dataset 1 (CCHIT)2) World-Vista	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
137	Improving trace accuracy through data-driven configuration and composition of tracing features	Transportation, Healthcare, Aerospace	3	Req	Code,Test,Design	2^6*3^5*36	University, Industrial, OSS	1) Industry 1 (Transport) 2) Links Industry 2 (Transport) 3) I-Trust (Health)4) CCHIT (Health) 5) E-Clinic (Health) 6) CM-1 (NASA)	6

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
139	A machine learning approach for tracing regulatory codes to product specific requirements	Healthcare	1	Req	HIPAA security rules	102392	OSS, University	1) Care2x 2) CCHIT 3) ClearHealth 4) Physician 5) iTrust 6) Trial Implementations 7) PatientOS 8) PracticeOne 9) Lauesen 10)World-	11

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
142	Recovery of Traceability Links between Software Documentation and Source Code	Development Library, Office Automation	2	Req	Code	19264	OSS,University	1) LEDA2) Albergate	2
143	Trust-Based Requirements Traceability	Communication	1	Req	Code	90478	OSS,University	1) Pooka 2) SIP	2
148	Supporting requirements to code traceability through refactoring	Healthcare, Entertainment, Office Automation	3	Req	Code	35224	OSS, Industrial, University	1) iTrust2) eTour3) Industrial software system WDS	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
150	Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited	Aerospace, Healthcare, Entertainment, Office Automation	4	Req	Code/design	70016	OSS, Industrial, University	1) iTrust2) eTour3) CM-14) Industrial software system WDS	4
154	An information theoretic approach for extracting and tracing non-functional requirements	Entertainment, Healthcare, Office Automation	3	Req	Code	155150	Unknown	1) SmartTrip2) SafeDrink3) BlueWallet	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
155	A Study on the Effect of Traceability Links in Software Maintenance	Office Automation	1	Test, Requirements, Design, Code	Build, Defect, Code,Test, Design, Code Inspection	226 + 1356 + 1017 + 171 + 152 + 16 + 16 + 96 = 3050 (File)	University	A room management system (RMS)	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
159	Enabling traceability reuse for impact analyses: A feasibility study in a safety context	Power and Automation	1	issue reports	Artifacts : Unique requirements, test case descriptions, user manuals, hardware, descriptions	29573880 (File)	Industrial	Issue Management System (IMS), from within a company in the power and automation domain, containing.	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
173	Code patterns for automatically validating requirements-to-code traces	Entertainment, Modeling, Software Eng. Tool	3	Req	Code	2197 + 3240 + 44047 + 37023 = 86507 (method)	OSS	1) VideoOn-Demand(VoD),2) Chess, 3) GanttProject,4) jHotDraw (JHD)	4

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
174	Effort and Quality of Recovering Requirements-to-Code Traces: Two Exploratory Experiments	Software Eng. Tool	1	Req	Code	1445 + 1968 + 13396 + 8704 = 25513	OSS	1) GranttProject2) ReactOS	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
176	Guidelines for Benchmarking Automated Software Traceability Techniques	Office Automation, Aerospace, Entertainment, Healthcare, Software Eng. Tool, Office Automation	6	artifact	artifact	935 + 1166 + 7700 + 2961 + 1410 + 6728 + 1173 + 6700 + 123424 = 152197 (File)	University, OSS	1) Albergate, 2) CM1,3) eAnci, 4) Easy Clinic, 5) Easy Clinic, 6) E-Tour, 7) Gantt, 8) SMOS,9) WV-CCHIT	9

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
177	Trace Matrix Analyzer (TMA)	Aerospace, Entertainment	2	artifact	artifact	931 + 50 = 981 (File)	University,OSS	1) MODIS2) Moon-Lander	2
178	How do we trace requirements: an initial study of analyst behavior in trace validation tasks	Software Eng. Tool	1	artifact	artifact	544 (File)	Industrial	ChangeStyle	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
180	Improving automated requirements trace retrieval: a study of term-based enhancement methods	Transportation, Software Eng. Tool, Power & Automation, Aerospace	4	Req	Java,UML classes	11644 + 2132 + 850 + 21850 + 51700 = 88176 (File)	Industrial, University	1) Ice Breaker System2) Event-Based Traceability3) Light Control System4) Light Control (LC) System5) CM1	5

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
181	Phrasing in Dynamic Requirements Trace Retrieva	Transportation, Software Eng. Tool, Power & Automation	3	Req	UML Diagram	11644 + 2132 + 850 = 14626	University	1) Ice Breaker System2) Event-Based Traceability3) Light Control System	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
182	The Detection and Classification of Non-Functional Requirements with Application to Early Aspects	Power and Automation	1	Documents	Documents	6156 + 333 = 6489 (File)	University, Industrial	1) 15 Requirement Documents2) Siemens IET document.	2
183	Using code ownership to improve IR-based Traceability Link Recovery	Entertainment, Office Automation	2	Req	Code	10092 + 6700 = 16792 (File)	University	1) eTour2)SMOS	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
185	When and How Using Structural Information to Improve IR-Based Traceability Recovery	Entertainment, Office Automation, Healthcare	3	Req	Code	1110 + 740 + 2331 + 10092 + 6700 = 20973 (File)	University	1) EasyClinic2) eTour3)SMOS	3
186	Traceability Recovery in RAD Software Systems	Power and Automation	1	Documents	Code	468 (File)	University	Transient Meter	1
187	Enhancing software artefact traceability recovery processes with link count information	Healthcare, Office Automation	2	Documents	Code	1410 + 1260 + 6700 + 9576 = 18946 (File)	University	1) EasyClinic2) SMOS	2

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
188	Recovering test-to-code traceability using slicing and textual analysis	Software Eng. Tool, Modeling	2	Test	Code	9568 + 63825 + 107250 + 59760 + 5916 = 246319 (File)	Industrial,OSS	1) SSAgileplanner2) Apache Ant 3) ArgoUML 4) Dependency Finder5) eXVantage	5

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
189	Applying a smoothing filter to improve IR-based traceability recovery processes: An empirical investigation	Entertainment, Communication, Aerospace, Healthcare	4	artifact	artifact	1410 + 940 + 2961 + 6728 + 931 + 2499 + 1551 = 17020 (File)	University, Industrial,OSS	1)eTour2)Pine3)MODIS i-Trust	5
190	Improving IR-based traceability recovery via noun-based indexing of software artifacts	Entertainment, Communication, Aerospace, Healthcare	4	artifact	artifact	1410 + 940 + 2961 + 6728 + 931 + 1166 + 2499 = 16635	University, Industrial,OSS	1)eTour2)Pine3)MODIS CM1	5

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
191	Evaluating test-to-code traceability recovery methods through controlled experiments	Software Eng. Tool, Modeling	2	Test	Code	220472	OSS, Industrial	1) AgilePlanner2) ArgoUML3) eXVantage	3
192	The role of artefact corpus in LSI-based traceability recovery	Healthcare, Office Automation, Entertainment	3	Documents	Code	1110 + 740 + 2331 = 4181 (File)	University	1) EasyClinic2) infoASL3) virtualRay4) SMOS5) eAnci	5

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
193	Improving IR-based Traceability Recovery Using Smoothing Filters	Healthcare, Communication	2	Documents	Code	124670 + 107835 + 71890 + 96831 = 401226 (File)	University,OSS	1) Easy-Clinic2) Pine	2
194	The role of the coverage analysis during IR-based traceability recovery: A controlled experiment	Healthcare	1	artifact	artifact	1110 + 1260 = 2370 (File)	University	EasyClinic	1
195	On the role of the nouns in IR-based traceability recovery	Healthcare	1	Req	Code	1110 + 740 + 2331 = 4181 (File)	University	EasyClinic	1

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
196	Traceability Recovery Using Numerical Analysis	Healthcare	1	Req	Code	1110 + 740 + 2331 = 4181 (File)	University	EasyClinic	1
197	Can Information Retrieval Techniques Effectively Support Traceability Link Recovery	Healthcare, Software Eng. Tool, Development Library	3	Req	Code	900 + 600 + 1890 + 1110 + 400 + 1260 + 740 + 3969 + 2331 + 1369 + 8004 + 19272 = 41845 (File)	University,OSS	1) 1 small Student SW2) 1 Medium Student SW - ADAMS3) 1 Large OSS - LEDA	3

Table A.1 continued from previous page

Num	Title	Domains	Num Domains	Source Artifacts	Target Artifacts	Trace Space	Type of Datasets	DataSet Name	Number of datasets
198	ADAMS Re-Trace: A Traceability Recovery Tool	Miscellenous	1	artifact	artifact	88 + 110 + 88 + 80 + 240 + 135 + 105 + 240 + 144 + 112 + 80 + 117 + 195 + 169 + 78 + 279 + 465 + 403 + 186 = 3314 (File)	University	7 Student Projects	7
201	Detecting, Tracing, and Monitoring Architectural Tactics in Code	Data Mgmt	1	Req	Code	17190	OSS	Apache Hadoop	1

Appendix B

DataSet Characteristics

Table B.1: DataSet Characteristics

ID	**Name**	**Description**	**Freq**	**Traceability Details**	**Trace Space**
1	Odyssey	a large-scale software development environment	1	Requirements To Code	56763
2	Mondrian	Online Analytical Processing server (OLAP). Allows business users to analyze large and complex amounts of data in real-time.	1	Tests To Classes	24231
3	AgilePlanner	Agileplanner : Industrial tool supports distributed agile teams in project planning	4	Tests To Classes	8073
4	JESSIE and JSSE	open-source implementations of theJava Secure Sockets Extension	1	Symbols To Classes	1330
5	Eclipse	Eclipse is Java Integrated Development Environment (IDE)	1	Issues To Methods	390978
6	Rhino	Rhino is an open-source implementation of JavaScript written entirely in Java. It is typically embedded into Java applications to provide scripting to end users.	2	Issues To Methods	76960
7	JDK 1.5	The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets.	1	Classes To Documents (PDF files)	45318
8	ArgoUML	Open source UML modeling tool.	4	Classes To Documents (Emails,	15989

Table B.1 continued from previous page

ID	**Name**	**Description**	**Freq**	**Traceability Details**	**Trace Space**
9	Freenet	Freenet is a peer-to-peer platform for censorship-resistant communication and publishing.	1	Classes To Documents (Emails)	192324
10	Jmeter	The Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.	1	Classes To Documents (Emails)	129456
11	Pine	an Open source email client	5	TeTotual Requirements To TeTotual use cases	2499
12	Pooka	Pooka is an email client written in Java using the JavaMail API.	3	Classes To Requirements	26820
13	jEdit	jEdit is a text editor for developers written in Java. jEdit includes a syntax highlighter that supports over 130 fileformats.	1	Classes To Requirements	16422
14	Joda Time	Joda Time is a Java library that aims to replace the Dateand Calendar Java API classes	1	Documents (Mailing List) To Source Code	3581
15	HttpComponents	HttpComponents is a Java library that simplifies the communication with a web server	1	Documents (API documentation) To Source Code	3581

Table B.1 continued from previous page

ID	**Name**	**Description**	**Freq**	**Traceability Details**	**Trace Space**
16	Hibernate	Hibernate is an Object-Relational Mapping framework(ORM) written in Java: it enables clients to persist objectsto a relational database	1	Documents (reference manual) To Source Code	3581
17	Xstream	XStream, is a Java library that enablesthe persistence of object graphs into XML files	1	Documents (Manual Documentation in HTML) To Source Code	3581
18	GanttProject	used to create Gantt charts and perform basic project management.	5	Requirements To Methods	44047
19	JHotDraw	JHotDraw is a Java GUI framework for technical and structured Graphics.	2	Requirements To Methods	37023
20	eXVantage	eXtreme Visual-Aid Novel Testing and Generation tools.	3	Methods To ClassesTest Classes To Classes	5916
21	ChangeStyle	a Java based style checker	3	High Level Requirements To Test Cases	527
22	ReactOS	OSS implementation of the Windows XP OS	1	Requirements To ClassesRequirements To Methods	59760

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
23	Event-Based Traceability (EBT) system.	represents a distributed traceability tool	3	Requirements To Classes	3240
24	SE450 Student Projects dataset	SE450 Student Projects dataset contains 15 anonymous student term projects for a MS level Software Engineering class.	1	Requirements To UML Classes-Requirements To JAVA Classes-Requirements To Low Level Requirements	88176
25	Apache Ant	Tool for automating software build processes	1	Methods To ClassesTest Classes To Classes	63825
26	Dependency Finder	A suite of tools for analyzing compiled Java code	1	Methods To ClassesTest Classes To Classes	59760
27	ADAMS	an artefact management system	1	Use Cases To Classes	8004
28	Apache Hadoop	The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models	1	Classes To Tactics	17190

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
29	LEDA	LEDA (Library of Efficient Data types and Algorithms), a well known library developed and distributed by Max-Planck Institut fiir Informatik, Saarbrticken, Germanyto-gether with its manual pages.	6	Terms To DocumentsClasses To PagesClasses To Requirements	
30	CCHIT	Reqs for Ambulatory, Health Info Exchange, EHR certifi-cation.	3	Requirements To Goals	2350
31	World- Vista	Open source version of Veteran Administrations HER	2	Requirements To Goals	17360
32	infoASL	Hospital management sysem composed of 93 use cases, 52 UML interaction diagrams, 28 test cases, and 20 code classes.	1	Code To UML Diagrams	740
33	iTrust	Open source HER North Carolina State University	7	UML Diagrams To Classes	940
34	Care2x	An open source Hospital Info. System	1	Regulatory Code To Require-ments	264
35	ClearHealth	Open source HER	1	Regulatory Code To Require-ments	484

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
36	Physician	Electronic exchange of info between clinicians.	1	Regulatory Code To Requirements	2205
37	PatientOS	Open source healthcare info. System	1	Regulatory Code To Requirements	1170
38	Trial Implementations	Priority info exchange. National Coord. for Health Info. technology (HIT)	1	Regulatory Code To Requirements	3100
39	PracticeOne	A Suite of healthcare info. Systems.	1	Regulatory Code To Requirements	238
40	Lauesen	Sample requirements specification for an HER	1	Regulatory Code To Requirements	1386
41	virtualRay	Medical records management system composed of 19 use cases, 79 test cases, and 65 code classes.	1	Code To Test Cases	2331
42	WV-CCHIT	developed by the Certification Commission for Healthcare Information Technology (CCHIT) and includes 235 requirements for managing electronic healthcare records.	1	Requirements To Regulatory Code	123424

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
43	SafeDrink	A software system designed to help users to manage their drinking habits.	1	Functional Requirements To Classes	56924
44	EasyClinic	is a system used to manage doctor offices appointments developed by students	20	Use Cases To Classes	1410
45	CM-1	consists of the entire requirement specification and the entire design document for a NASA scientific instrument.	12	Requirements To Design Documents	1166
46	MODIS	MODIS dataset consists of 19 high level and 49 low-level requirements for open source ModerateResolution Imaging Spectroradiometer (MODIS) developed byNASA.	10	High Level Requirements To Low Level Requirements	931
47	Chess	game software	1	Requirements To Methods	3240
48	Albergate (Hotel Management System)	Albergate, is a softwaresystem designed to implement all the operations required toadministrate and manage a small/medium size hotel (roomreservation, bill calculation, etc.)	6	Classes To Functional Requirements	960
49	CoffeeMaker	coffee maker software	1	Requirements To Methods	272

Table B.1 continued from previous page

ID	**Name**	**Description**	**Freq**	**Traceability Details**	**Trace Space**
50	Video on Demand	This software system isessentially a movie player that can search for movies, select and play them.	3	Requirements To Methods	2197
51	Project 10 (P10)	specifies requirements for an online version of a game like Battleship.	1	None Functional Requirements To Functional Requirements	570
52	MoonLander	text based Student Software consisting of 10 high level requirements and 5 test cases.	1	High Level Requirements To Test Cases	50
53	Ice Breaker System (IBS)	IBS manages deicing services to prevent ice formation on roads	4	Requirements To UML Diagrams	11644
54	Vehicle Highway System	Vehicle highway system	1	Requirements To Design Documents	359566
55	Driver-Optional Highway System (DOHS)	The DOHS provides environmentalcontrols used by both manned and unmanned vehicles to propose routing, to prevent vehicles from enteringaccident zones, and to provide a wide range of directives.	2	Requirements To Design Documents	233046 + 561429 = 794475
56	Industry 1	transportation	1	Requirements To Design Documents	1371968

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
57	Links Industry 2	transportation	1	Requirements To Design Documents	211680
58	wiper control system for a car	wiper control system for a car, created for Volkswagen AG.	1	UML Diagrams To Code	N/A
59	SIP Communicator	audio/video Internet phone and instant messenger.	2	Classes To Requirements	145222
60	GSM 11-11 technical specification	The GSM 11-11 document defines the security features, the interface functions, the contents of the files requiredfor the GSM applications and the commands available on the SIM-card.	1	Requirements To Test Cases	42
61	Issue Management System (IMS)	4845 issue reports containing Impact Analysis Information.	1	Issues To None Code Artifacts	29573880
62	Siemens IET document	An integrated engineering toolset (IET) under development at Siemens Logistics and Automation plant.	1	Pages To None Functional Requirements	333

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
63	Transient Meter	Transient Meter is a distributed measurement systemfor power quality monitoring developed at the Measurement-Laboratory of University of Sannio, and currentlyunder beta-test by the Italian power agency.	1	Requirements To Classes	468
64	WDS (workforce development)	A proprietary software-intensive platform that provides technological solutions for service delivery and workforce development in a specific region of the United States.	1	Requirements To Classes	2500
65	BlueWallet	A subscription-based web service that provides users with options to plan their budgets and manage their personal finances.	1	Functional Requirements To Classes	56924
66	Light Control system (LC).	constructed from a cases study of the lighting system	3	Requirements To Classes	21850
67	eTour	eTour is an electronic tourist guide developed by students.	10	Use Cases To Code	6728

Table B.1 continued from previous page

ID	**Name**	**Description**	**Freq**	**Traceability Details**	**Trace Space**
68	A room management system (RMS)	created by students.Main features are reserving rooms, displaying reservations and editing reservations.	1	Tests To CodeRequirements To TestRequirements To Design DocumentsDesign Documents To CodeCode To Code	226 + 1356 + 1017 + 171 + 152 + 16 + 16 + 96 = 3050
69	SmartTrip	An Android mobile application that is designed to provide up-to-date routing and pricing information for users planning road trips.	1	Functional Requirements To Classes	29410
70	eAnci	A system providing support to manage Italian municipalities	3	Use Cases To Classes	7645
71	Library management system	library management system developed by students of the Technical University of Ilmenau.	1	UML Diagrams To Code	N/A
72	SMOS	SMOS is an application that is used to monitor high school students (e.g., absence, grades).	7	Use Cases To Classes	6700

Table B.1 continued from previous page

ID	Name	Description	Freq	Traceability Details	Trace Space
73	7 Student Projects	University student projects	1	Use Cases To Module DiagramUse Cases To Dynamic DiagramUse Cases To Test CasesRequirements To Use CasesRequirements To Module DescriptionRequirements To Dynamic DiagramRequirements To Test CasesUse Cases To RequirementsUse Cases To Module DiagramUse Cases To Dynamic DiagramUse Cases To Test CasesDynamic Diagram To Use CasesDynamic Diagram To Module DiagramDynamic Diagram To Dynamic DiagramDynamic Diagram To Test Cases	88 + 110 + 88 + 80 + 240 + 135 + 105 + 240 + 144 + 112 + 80 + 117 + 195 + 169 + 78 + 279 + 465 + 403 + 186 = 3314

Bibliography

[1] N. Ali, Y. Gueheneuc, and G. Antoniol. Trust-based requirements traceability. In *ICPC*, pages 111–120, June 2011.

[2] N. Ali, Y. Gueheneuc, and G. Antoniol. Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. *TSE*, 39(5):725–741, May 2013.

[3] N. Ali, Z. Sharafl, Y. Gueheneuc, and G. Antoniol. An empirical study on requirements traceability using eye-tracking. In *ICSM*, pages 191–200, 2012.

[4] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. Information retrieval models for recovering traceability links between code and documentation. In *ICSM*, pages 40–49, 2000.

[5] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. *TSE*, 28(10):970–983, Oct 2002.

[6] G. Bavota, A. De Lucia, R. Oliveto, A. Panichella, F. Ricci, and G. Tortora. The role of artefact corpus in lsi-based traceability recovery. In *TEFSE*, pages 83–89, 2013.

[7] G. Bavota, A. De Lucia, R. Oliveto, and G. Tortora. Enhancing software artefact traceability recovery processes with link count information. *Information and Software Technology*, 56(2):163–182, 2014.

[8] M. Borg, O.C.Z. Gotel, and K. Wnuk. Enabling traceability reuse for impact analyses: A feasibility study in a safety context. In *TEFSE*, pages 72–78, 2013.

[9] Markus Borg, Per Runeson, and Anders Ardö. Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. *EMSE*, 19(6):1565–1616, 2014.

[10] F. Bouquet, E. Jaffuel, B. Legeard, F. Peureux, and M. Utting. Requirements traceability in automated test generation: Application to smart card software validation. *SIGSOFT Softw. Eng. Notes*, 30(4):1–7, May 2005.

[11] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *JSS*, 80(4):571 – 583, 2007.

[12] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella. On the role of the nouns in ir-based traceability recovery. In *ICPC*, pages 148–157, 2009.

[13] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella. Traceability recovery using numerical analysis. In *WCRE*, pages 195–204, Oct 2009.

[14] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella. Improving ir-based traceability recovery via noun-based indexing of software artifacts. *SEP*, 25(7):743–762, 2013.

[15] Xiaofan Chen and John Grundy. Improving automated documentation to code traceability by combining retrieval techniques. In *ASE*, pages 223–232, 2011.

[16] J. Cleland-Huang, R. Settimi, Chuan Duan, and Xuchang Zou. Utilizing supporting evidence to improve dynamic requirements traceability. In *RE*, pages 135–144, 2005.

[17] J. Cleland-Huang, R. Settimi, Xuchang Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *RE*, pages 39–48, 2006.

[18] Jane Cleland-Huang, Adam Czauderna, Marek Gibiec, and John Emenecker. A machine learning approach for tracing regulatory codes to product specific requirements. In *ICSE - Volume 1*, pages 155–164, 2010.

[19] Jane Cleland-Huang, Orlena CZ Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. Software traceability: trends and future directions. In *Future of Software Engineering*, pages 55–69, 2014.

[20] Jane Cleland-Huang, Raffaella Settimi, Oussama BenKhadra, Eugenia Berezhanskaya, and Selvia Christina. Goal-centric traceability for managing non-functional requirements. In *ICSE*, pages 362–371, 2005.

[21] Christopher S. Corley, Nicholas A. Kraft, Letha H. Etzkorn, and Stacy K. Lukins. Recovering traceability links between source code and fixed bugs via patch analysis. In *TEFSE*, pages 31–37, 2011.

[22] Barthélémy Dagenais and Martin P. Robillard. Recovering traceability links between an api and its learning resources. In *ICSE*, 2012.

[23] T. Dasgupta, M. Grechanik, E. Moritz, B. Dit, and D. Poshyvanyk. Enhancing software traceability by automatically expanding corpora with relevant documentation. In *ICSM*, pages 320–329, 2013.

[24] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella. Improving ir-based traceability recovery using smoothing filters. In *ICPC*, pages 21–30, 2011.

[25] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Enhancing an artefact management system with traceability recovery features. In *ICSM*, pages 306–315, 2004.

[26] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Adams re-trace: A traceability recovery tool. In *CSMR*, pages 32–41, 2005.

[27] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Can information retrieval techniques effectively support traceability link recovery? In *ICPC*, pages 307–316, 2006.

[28] A. De Lucia, R. Oliveto, and P. Sgueglia. Incremental approach and user feedbacks: a silver bullet for traceability recovery. In *ICSM*, pages 299–309, 2006.

[29] A. De Lucia, R. Oliveto, and G. Tortora. IR-based traceability recovery processes: An empirical comparison of "one-shot" and incremental processes. In *ASE*, pages 39–48, 2008.

[30] A. De Lucia, R. Oliveto, and G. Tortora. The role of the coverage analysis during ir-based traceability recovery: A controlled experiment. In *ICSM*, pages 371–380, 2009.

[31] Andrea De Lucia, Rocco Oliveto, and Genoveffa Tortora. Assessing ir-based traceability recovery tools through controlled experiments. *EMSE*, 14(1):57–92, 2009.

[32] M. Di Penta, S. Gradara, and G. Antoniol. Traceability recovery in rad software systems. In *Int. Workshop on Program Comprehension (IWPC)*, pages 207–216, 2002.

[33] D. Diaz, G. Bavota, A. Marcus, R. Oliveto, S. Takahashi, and A. De Lucia. Using code ownership to improve ir-based traceability link recovery. In *ICPC*, pages 123–132, 2013.

[34] A. Egyed, F. Graf, and P. Grunbacher. Effort and quality of recovering requirements-to-code traces: Two exploratory experiments. In *RE*, pages 221–230, 2010.

[35] A. Egyed and P. Grunbacher. Automating requirements traceability: Beyond the record replay paradigm. In *ASE*, pages 163–171, 2002.

[36] V. Gervasi and D. Zowghi. Supporting traceability through affinity mining. In *RE*, pages 143–152, 2014.

[37] M. Gethers, R. Oliveto, D. Poshyvanyk, and A.D. Lucia. On integrating orthogonal information retrieval methods to improve traceability recovery. In *ICSM*, pages 133–142, 2011.

[38] A. Ghabi and A. Egyed. Code patterns for automatically validating requirements-to-code traces. In *ASE*, pages 200–209, 2012.

[39] Jin Guo, J. Cleland-Huang, and B. Berenbach. Foundations for an expert system in domain-specific traceability. In *RE*, pages 42–51, 2013.

[40] Jin Guo, Natawut Monaikul, Cody Plepel, and Jane Cleland-Huang. Towards an intelligent domain-specific traceability solution. In *ASE*, pages 755–766, 2014.

[41] J. H. Hayes, G. Antoniol, B. Adams, and Y. G. Guhneuc. Inherent characteristics of traceability artifacts less is more. In *RE*, pages 196–201, 2015.

[42] J.H. Hayes, A. Dekhtyar, and J. Osborne. Improving requirements tracing via information retrieval. In *RE*, pages 138–147, 2003.

[43] J.H. Hayes, A. Dekhtyar, S.K. Sundaram, and S. Howard. Helping analysts trace requirements: an objective look. In *RE*, pages 249–259, 2004.

[44] E.A. Holbrook, J.H. Hayes, and A. Dekhtyar. Toward automating requirements satisfaction assessment. In *RE*, pages 149–158, 2009.

[45] K. Jaber, B. Sharif, and Chang Liu. A study on the effect of traceability links in software maintenance. *Access, IEEE*, 1:726–741, 2013.

[46] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, EBSE Technical Report EBSE-2007-01, 2007.

[47] W. K. Kong and J. H. Hayes. Proximity-based traceability: An empirical validation using ranked retrieval and set-based measures. In *Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 45–52, 2011.

[48] W.-K. Kong, J. Huffman Hayes, A. Dekhtyar, and J. Holden. How do we trace requirements: An initial study of analyst behavior in trace validation tasks. In *Int. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 32–39, 2011.

[49] Hongyu Kuang, P. Mader, Hao Hu, A. Ghabi, Liguo Huang, Lv Jian, and A. Egyed. Do data dependencies in source code complement call dependencies for understanding requirements traceability? In *ICSM*, pages 181–190, 2012.

[50] Wenbin Li, J.H. Hayes, Fan Yang, K. Imai, J. Yannelli, C. Carnes, and M. Doyle. Trace matrix analyzer (tma). In *TEFSE*, pages 44–50, 2013.

[51] Yonghua Li and J. Cleland-Huang. Ontology-based trace retrieval. In *TEFSE*, pages 30–36, 2013.

[52] Gernot A. Liebchen and Martin Shepperd. Data sets and data quality in software engineering. In *Int. Workshop on Predictor Models in Software Engineering (PROMISE)*, pages 39–44, 2008.

[53] Sugandha Lohar, Sorawit Amornborvornwong, Andrea Zisman, and Jane Cleland-Huang. Improving trace accuracy through data-driven configuration and composition of tracing features. In *ESEC/FSE*, pages 378–388. ACM, 2013.

[54] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella. Applying a smoothing filter to improve ir-based traceability recovery processes: An empirical investigation. *Information and Software Technology*, 55(4):741–754, 2013.

[55] Andrea De Lucia, Fausto Fasano, Rocco Oliveto, and Genoveffa Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *TOSEM*, 16(4), September 2007.

[56] P. Mader, O. Gotel, and I. Philippow. Rule-based maintenance of post-requirements traceability relations. In *RE*, pages 23–32, 2008.

[57] A. Mahmoud. An information theoretic approach for extracting and tracing non-functional requirements. In *RE*, pages 36–45, 2015.

[58] A. Mahmoud and N. Niu. Source code indexing for automated tracing. In *TEFSE*, pages 3–9, 2011.

[59] A. Mahmoud and N. Niu. Supporting requirements to code traceability through refactoring. *REJ*, 19(3):309–329, 2013.

[60] A. Marcus and J. I. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *ICSE*, pages 125–135, 2003.

[61] Collin McMillan, Denys Poshyvanyk, and Meghan Revelle. Combining textual and structural analysis of software artifacts for traceability link recovery. In *TEFSE*, pages 41–48, 2009.

[62] M. Mirakhorli and J. Cleland-Huang. Detecting, tracing, and monitoring architectural tactics in code. *TSE*, 42(3):205–220, March 2016.

[63] L.G.P. Murta, A. Van Der Hoek, and C.M.L. Werner. Archtrace: Policy-based support for managing evolving architecture-to-implementation traceability links. In *ASE*, pages 135–144, 2006.

[64] Sunil Nair, Jose Luis De La Vara, and Sagar Sen. A review of traceability research at the requirements engineering conference re@ 21. In *RE*, pages 222–229, 2013.

[65] N. Niu and A. Mahmoud. Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited. In *RE*, pages 81–90, 2012.

[66] Nan Niu, T. Bhowmik, Hui Liu, and Zhendong Niu. Traceability-enabled refactoring for managing just-in-time requirements. In *RE*, pages 133–142, 2014.

[67] S. Pandanaboyana, S. Sridharan, J. Yannelli, and J.H. Hayes. Requirements tracing on target (retro) enhanced with an automated thesaurus builder: An empirical study. In *TEFSE*, 2013.

[68] A. Panichella, A. De Lucia, and A. Zaidman. Adaptive user feedback for ir-based traceability recovery. In *SST*, pages 15–21, 2015.

[69] A. Panichella, C. McMillan, E. Moritz, D. Palmieri, R. Oliveto, D. Poshyvanyk, and A. De Lucia. When and how using structural information to improve ir-based traceability recovery. In *CSMR*, pages 199–208, 2013.

[70] D. Port, A. Nikora, J. H. Hayes, and L. Huang. Text mining support for software requirements: Traceability assurance. In *Hawaii Int. Conf. on System Sciences*, pages 1–11, 2011.

[71] A. Qusef, G. Bavota, R. Oliveto, A. De Lucia, and D. Binkley. Scotch: Test-to-code traceability using slicing and conceptual coupling. In *ICSM*, pages 63–72, 2011.

[72] A. Qusef, G. Bavota, R. Oliveto, Andrea De Lucia, and D. Binkley. Evaluating test-to-code traceability recovery methods through controlled experiments. *SEP*, (11):1167–1191, 2013.

[73] A. Qusef, G. Bavota, R. Oliveto, A. De Lucia, and D. Binkley. Recovering test-to-code traceability using slicing and textual analysis. *JSS*, 88:147–168, 2014.

[74] A. Qusef, R. Oliveto, and A. De Lucia. Recovering traceability links between unit tests and classes under test: An improved method. In *ICSM*, pages 1–10, 2010.

[75] M. Rahimi, M. Mirakhorli, and J. Cleland-Huang. Automated extraction and visualization of quality concerns from requirements specifications. In *RE*, pages 253–262, 2014.

[76] Iván Santiago, Alvaro Jiménez, Juan Manuel Vara, Valeria De Castro, Verónica A Bollati, and Esperanza Marcos. Model-driven engineering as a new landscape for traceability management: A systematic literature review. *Information and Software Technology*, 54(12):1340–1356, 2012.

[77] M. Shepperd, Q. Song, Z. Sun, and C. Mair. Data quality: Some comments on the nasa software defect datasets. *TSE*, 39(9):1208–1215, Sept 2013.

[78] Yonghee Shin, J.H. Hayes, and J. Cleland-Huang. Guidelines for benchmarking automated software traceability techniques. In *SST*, pages 61–67, 2015.

[79] H. Sultanov and J.H. Hayes. Application of reinforcement learning to requirements engineering: requirements tracing. In *RE*, pages 52–61, 2013.

[80] Senthil Karthikeyan Sundaram, Jane Huffman Hayes, and Alexander Dekhtyar. Baselines in requirements tracing. In *Workshop on Predictor Models in Software Engineering (PROMISE)*, pages 1–6, 2005.

[81] Z. Xuchang, R. Settimi, and J. Cleland-Huang. Phrasing in dynamic requirements trace retrieval. In *COMPSAC*, volume 1, pages 265–272, 2006.

[82] Suresh Yadla, Huffman Jane Hayes, and Alex Dekhtyar. Tracing requirements to defect reports: an application of information retrieval techniques. *ISSE*, 1(2):116–124, 2005.

[83] Hsin yi Jiang, T.N. Nguyen, Ing-Xiang Chen, H. Jaygarl, and C.K. Chang. Incremental latent semantic indexing for automatic traceability link evolution management. In *ASE*, pages 59–68, 2008.

[84] Yijun Yu, Jan Jurjens, and J. Mylopoulos. Traceability for the maintenance of secure software. In *ICSM*, pages 297–306, 2008.

[85] He Zhang, Muhammad Ali Babar, and Paolo Tell. Identifying relevant studies in software engineering. *Information and Software Technology*, 53(6):625–637, 2011.

[86] X. Zou, R. Settimi, and J. Cleland-Huang. Improving automated requirements trace retrieval: a study of term-based enhancement methods. *EMSE*, 15(2):119–146, 2009.